Journal of the Korea Academia-Industrial cooperation Society
Vol. 15, No. 1 pp. 441-448, 2014

http://dx.doi.org/10.5762/KAIS.2014.15.1.441

USN 멀티플랫폼을 위한 노드 소프트웨어 개발을 지원하는 속성 설계 기법

이우진[1], 최일우[2], 김주일[3*]
[1]세종대학교 정보통신공학과, [2]강남대학교 교양학부, [3]지아이티 GDS진단팀

The Attributes Design Technique to Support Node Software Development for USN Multi-Platform

Woo-Jin Lee[1], Il-Woo Choi[2] and Ju-Il Kim[3*]
[1]**Dept. of Information and Communication Engineering, Sejong University**
[2]**Division of General Studies, Kangnam University**
[3]**GDS Diagnostic Team, GIT**

요 약 USN(Ubiquitous Sensor Network) 응용 소프트웨어는 다양한 대상 운영체제의 핵심모듈들을 기반으로, 다양한 종류의 센서 노드들을 유기적으로 제어하는 복잡한 특징을 가진다. 이에 따라, 현재 USN 응용 분야에서는 효율적으로 소프트웨어를 개발하기 위한 다양한 연구가 진행되고 있다. 본 논문에서는 센서 네트워크 환경에서 하나의 모델로부터 여러 플랫폼에 맞는 노드 소프트웨어를 효율적으로 개발하기 위한 속성 기반 개발을 지원하는 속성 설계 기법을 제시한다. 제시된 기법에서는 Platform Independent Model과 Platform Specific Model을 만들기 위한 속성을 설계하는 방법을 기술한다. 본 논문에서 제시하는 기법을 사용하면, 속성 값의 선택을 통하여 멀티 플랫폼을 위한 노드 소프트웨어를 손쉽게 디자인함으로써 소프트웨어 개발 생산성은 증대된다. 또한 운영체제의 변화에 따라 속성 변경을 통하여 노드 소프트웨어를 쉽게 재생성할 수 있으므로, 소프트웨어의 유지보수성이 향상된다.

Abstract USN(Ubiquitous Sensor Network) application software has a characteristic that it controls a variety of sensor nodes based on the various target operating systems. Accordingly, many researches for efficient development of USN application software are being performed. In this paper, the attributes design technique to support attribute-based development of USN node software for multi-platform is proposed. In the proposed technique, the method to design attributes for modeling Platform Independent Model and Platform Specific Model is presented. When using the proposed technique, productivity of software development will be increased because node software design for multi-platform is easily performed by selecting values of attributes. Also, maintainability of software will be increased because node software is easily regenerated by changing attributes according to the changes of operating systems.

Key Words : Attributes design, Multi-platform, Node software, Ubiquitous sensor network

1. 서론

센서 네트워크[1]는 저전력, 초경량의 센서들로 구성된 네트워크로 사물과 주변 환경을 감지하여 사용자에게 정보를 전달해 줌으로써, 사용자들은 센서 네트워크를 통하여 언제, 어디서나 정보를 전달받을 수 있는 USN(Ubiquitous Sensor Network)환경을 구축할 수 있으며 다양한 분야에서 연구, 개발되어지고 있다.

센서 네트워크를 구성하는 노드들은 저전력, 초경량으로 여러 가지 제약사항을 가지므로 노드 소프트웨어는

*Corresponding Author : Ju-Il Kim(GIT)
Tel: +82-10-3473-0382 email: sespop@empal.com
Received July 31, 2013 Revised (1st September 24, 2013, 2nd November 1, 2013) Accepted January 9, 2014

이러한 제약사항을 고려하여 개발해야 한다. 또한 노드 소프트웨어는 운영체제를 기반으로 개발되는데 센서 네트워크 운영체제는 여러 종류가 존재하므로, 노드 소프트웨어는 타겟 운영체제의 종류 또한 고려되어야 한다.

이와 같이 여러 제약사항 및 다양한 운영체제를 고려하여 소프트웨어를 개발하는 것은 매우 번거로운 일이다. 이를 극복하기위한 여러 방법들[2-6]이 연구되어있으나, 기존 방법은 여러 운영체제에 따라 각기 서로 다른 기법을 사용하여야 하므로 효율적이지 못하다.

본 논문에서는 플랫폼에 따라 서로 다른 기법을 통하여 응용 소프트웨어를 개발하는 기존 방법들의 한계점을 극복하기 위하여, 하나의 모델로부터 여러 플랫폼에 맞는 응용 소프트웨어를 개발하기 위한 MDA(Model Driven Architecture)와 응용 소프트웨어를 편리하게 개발하기 위한 속성 기반 개발 개념을 적용한 개발 방법을 지원하는 속성 설계 기법을 제시한다.

2. 관련 연구

2.1 센서 네트워크 운영체제 플랫폼

현 센서 네트워크 운영체제 플랫폼은 센서 노드가 가지는 자원의 제약을 극복하고 다양한 분야에서의 응용 소프트웨어에 대한 개발을 지원하며, 운영체제가 가지는 전반적인 기능을 포함하도록 발전하였고, 그 종류 또한 다양해졌다.

이러한 센서 네트워크 운영체제 플랫폼은 센서 네트워크 응용 소프트웨어를 개발하는데 매우 중요한 역할을 한다. 센서 네트워크 응용 소프트웨어는 운영체제의 모듈과 운영체제의 모듈을 사용하는 응용 코드가 합쳐져서 개발된다. 따라서 센서 네트워크 응용 소프트웨어를 개발하는 경우에는 타겟 운영체제 플랫폼이 제공하는 기능을 포함하도록 설계하고 코드를 생성해야 한다.

센서 네트워크 응용 소프트웨어 개발에 사용되는 주요 운영체제 플랫폼으로는 Nano-Qplus[7], TinyOS[8], SOS[9], MANTIS[10], Contiki[11], t-kernel[12], LiteOS[13], Nano-RK[14] 등이 있다.

2.2 MDA

MDA(Model Driven Architecture)[15]는 지속적으로 변화하는 기술 상황에 따라 시스템을 기술 변화에 맞게 통합, 변화, 유지하는 문제를 해결하기 위하여 OMG (Object Management Group)가 제시한 소프트웨어 아키텍처이다. MDA는 시스템의 설계와 명세를 정형화된 모델로 기술 플랫폼과 분리하여 기술하도록 하며, 실제 구현과 관련된 모델은 매핑을 통해서 기술 플랫폼에 독립적으로 기술된 모델을 변환하여 생성하도록 한다. MDA 기반 개발 방법에서는 개발하고자 하는 시스템에 대한 PIM(Platform Independent Model)을 작성하고, PIM을 바탕으로 특정 플랫폼에 특화된 PSM(Platform Specific Model)을 생성하여 PSM으로부터 해당 플랫폼을 기반으로 하는 응용 소프트웨어를 구현할 수 있도록 한다.

이와 같은 특징을 가지는 MDA 기반 개발 방법을 이용하여 응용 시스템을 개발하면, 기반 기술이 변화하더라도 PIM 변환을 통해 해당 기술변화에 대응하는 PSM을 생성함으로써 시스템을 보다 효율적으로 유지할 수 있다. 이렇듯 하나의 시스템을 PIM과 PSM으로 기술하는 것은 좀 더 유연하고 생산성 높은 모델 수준의 시스템 통합을 가능하게 한다.

본 논문에서는 멀티플랫폼을 위한 센서 네트워크 응용 소프트웨어를 개발하기 위하여 MDA 개발 개념을 적용한다.

2.3 속성 기반 프로그래밍

속성 기반 프로그래밍은 응용 소프트웨어 코드 작성 시에 속성을 이용하여 meta-data를 추가하여, 실행 시에 동적으로 meta-data의 값을 변경하도록 함으로써 코드를 직접 변경하거나 재컴파일을 하지 않고 응용 소프트웨어를 재구성할 수 있도록 하는 프로그래밍 방법이다[16]. 또한 프로그램의 추상화 수준을 높이는 방법으로, 개발자들이 속성을 이용하여 프로그램을 작성하면 구현 시에 속성의 값에 따라 프로그램 코드가 삽입되도록 함으로써 개발자들이 쉽게 프로그램을 작성하도록 한다[17]. Visual C++에서도 이러한 속성 기반 프로그래밍 기법을 이용하여 생성된 Object 파일에 코드를 삽입하거나 변경할 수 있도록 한다[18]. 센서 네트워크 응용 소프트웨어 개발 기법인 SPIDEY[3]에서는 논리적인 이웃을 정의하기 위하여 속성 기반 프로그래밍 방법을 사용하기도 한다.

그러나 속성 기반의 프로그래밍 방법에서는 속성과 그 값을 프로그램 작성 시 직접 넣어줘야 하므로 속성이 많아지는 경우에는 프로그램의 복잡성이 증가한다는 단점이 있다.

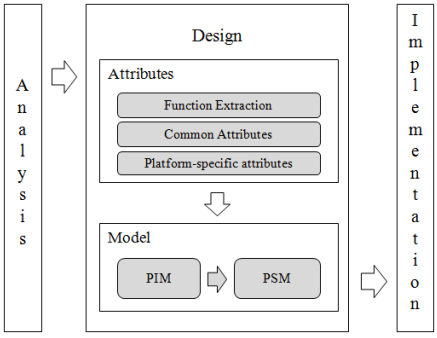
본 논문에서는 이러한 속성 기반 프로그래밍 방법이 가지는 단점을 보완하여 좀 더 쉽게 속성을 활용할 수 있는 방법을 제시한다. 본 논문에서 적용한 속성 기반 개발 방법은 기존의 속성 기반 프로그래밍과는 달리 센서 운영체제가 제공하는 기능과 연결된 속성을 미리 정의하여, 개발자가 프로그램 작성 시에 속성과 그 값을 직접 넣어

주는 것이 아니라, 노드 소프트웨어 설계 시에 미리 정의된 속성에 대한 값을 선택하기만 하면 그 값에 따라 운영체제가 제공하는 모듈이 선택되고 소프트웨어 코드가 자동으로 생성되도록 한다.

3. 속성 설계

본 논문에서 제안하는 속성 설계 기법은 멀티플랫폼을 위한 센서 네트워크 응용 소프트웨어를 효율적으로 개발할 수 있는 개발 방법을 지원한다.

Fig. 1은 멀티플랫폼을 위한 속성 기반의 센서 네트워크 응용 소프트웨어 개발 방법의 개념을 나타낸다. 센서 네트워크를 위한 운영체제 플랫폼은 매우 다양하므로 개발자는 플랫폼에 독립적인 모델인 PIM을 설계하고, PIM은 플랫폼 및 노드의 특성에 따른 PSM으로 변환되어야 한다. 이 과정에서 미리 정의한 노드 특성에 따른 독립적 혹은 종속적 속성 값을 선택하고 이 속성을 기반으로 PIM모델을 플랫폼 특성을 반영한 PSM모델로 변환되도록 지원한다.



[Fig. 1] The concept of the attribute-based sensor network software development for multi-platform

속성을 기반으로 센서 네트워크 응용 소프트웨어를 설계하므로, 이를 위한 속성을 설계해야 한다. 속성 설계시 먼저 운영체제 플랫폼의 기능을 도출하고, 도출된 기능을 바탕으로 센서 네트워크 응용 소프트웨어의 설계를 위한 공통속성과 종속속성을 설계한다.

3.1 플랫폼 기능 도출

기능 도출은 개발할 센서 네트워크 응용 소프트웨어가 수행될 주요 운영체제 플랫폼들이 제공하는 기능을 도출하는 것으로, 이러한 기능은 센서 네트워크 응용 소프트웨어의 개발을 위한 속성을 설계하기 위해 필요하다.

기능 도출은 여러 운영체제 플랫폼들이 공통적으로 제공하는 기능과 특정 운영체제 플랫폼에서만 제공하는 운영체제 종속적인 기능을 분리하여 도출한다. 플랫폼 독립적인 기능과 플랫폼 종속적인 기능은 Table 1과 같은 운영체제의 종류와 기능에 대한 매트릭스를 이용하여 도출한다.

[Table 1] Platform function extraction matrix

Function \ Platform	OS #1	OS #2	OS #3	OS #4	OS #n	
Function #1	√	√	√			*Fs*
Function #2	√				√	*Fs*
Function #3	√	√	√	√	√	*Fi*
Function #4	√			√		*Fs*
Function #n	√	√	√	√	√	*Fi*

Fi = OSf1 ∩ OSf2 ∩ OSf3 .... ∩ OSfn
Fi: Common function
Fs: Platform-specific function

3.2 공통속성 설계

공통속성 설계의 목적은 센서 네트워크 응용 소프트웨어의 PIM을 설계하는데 필요한 속성을 설계하기 위한 것으로, 여러 운영체제들이 제공하는 공통된 기능을 응용 소프트웨어에 포함시키기 위해 필요한 속성과 플랫폼과 상관없이 센서 네트워크 응용 소프트웨어의 개발을 위해 필요한 요소를 설정하기 위한 속성을 설계한다. 이러한 공통속성은 센서 네트워크 응용 소프트웨어를 생성하기 위해서 필요한 기본적인 속성이라고 할 수 있다.

센서 네트워크 응용 소프트웨어의 공통된 기능을 설정하기 위한 속성은 플랫폼 기능 도출 기법을 통하여 도출한 공통된 기능을 바탕으로 설계한다. 또한 플랫폼과 상관없이 센서 네트워크 응용 소프트웨어의 개발을 위해 공통적으로 필요한 요소들을 설정하기 위한 속성을 생성하기 위해서는 운영체제의 기능과 상관없이 응용 소프트웨어의 설계를 위해 필요한 요소들을 찾아내고, 각각의 요소를 표현하기 위한 속성을 정의한다. 공통속성 설계 방법을 정리하면 다음과 같다.

CA.1 운영체제의 기능을 선택하기 위한 공통속성은 운영체제들의 공통기능으로부터 도출

CA.2 해당 기능을 포함할 것인지, 포함하지 않을 것인지를 결정하기 위한 속성은 '기능이름_Enable'과 같은 명칭으로 정의

[*예) 'Timer 기능'의 포함 여부를 결정하기 위한 속성은 'Timer_Enable'로 정의*]

CA.3 해당 기능에 대하여 선택 옵션이 있는 경우, 속성은 '기능이름' 으로 명칭을 정의, 옵션을 속성의 선택 값으로 정의

CA.3.1 옵션을 선택하지 않아도 되는 경우, ‘none’ 옵션을 정의

CA.3.2 옵션을 여러 가지 선택해도 되는 경우, 선택 가능한 모든 경우를 옵션으로 정의

[*예) printf와 scanf의 UART 기능 중에서 필요한 것을 사용하거나 필요하지 않으면 사용하지 않아도 되는 경우, 속성명을 UART로, 옵션은 none, printf, scanf, printf&scanf로 정의*]

CA.4 센서 네트워크 응용 소프트웨어의 개발을 위해 공통적으로 필요한 요소의 값을 설정하기 위해서는 이를 위한 공통속성을 정의. 속성의 명칭은 개발 요소의 의미를 명확히 알 수 있도록 정의

[*예) 노드의 종류를 설정하기 위한 속성은 nodeType으로, 노드의 ID를 설정하기 위한 속성은 nodeID로 정의*]

CA.5 각 속성의 유형을 결정

ADC: 데이터 감지와 관련된 속성 유형

PSS: 플랫폼에 따라 옵션이 달라지는 속성 유형

COM: ADC, PSS를 제외한 공통속성 유형

3.3 종속속성 설계

종속속성의 설계의 목적은 센서 네트워크 응용 소프트웨어의 PSM 작성을 위한 속성을 설계하는 것으로, 각 운영체제에 종속된 기능을 응용 소프트웨어에 포함시키기 위해 필요한 속성과 센서 네트워크 응용 소프트웨어의 개발을 위해 필요한 운영체제 종속적인 요소를 설정하기 위한 속성을 설계한다.

센서 네트워크 응용 소프트웨어의 플랫폼 종속적인 기능을 설정하기 위한 속성은 플랫폼 기능 도출 기법을 통하여 도출한 플랫폼 종속적인 기능을 바탕으로 설계한다. 또한 센서 네트워크 응용 소프트웨어의 개발을 위해 필요한 운영체제 종속적인 요소들을 설정하기 위한 속성을 생성하기 위해서는 운영체제 종속적인 프로그램 개발 요소를 찾아내고, 각각의 요소를 표현하기 위한 속성을 정의한다. 플랫폼 종속적인 속성의 설계 방법을 정리하면 다음과 같다.

SA.1 특정 운영체제의 기능을 선택하기 위한 종속속성은 해당 운영체제의 기능으로부터 도출

SA.2 해당 기능을 포함할 것인지, 포함하지 않을 것인지를 결정하기 위한 속성은 ‘기능이름_Enable’과 같은 명칭으로 정의 (CA.2와 같은 방법)

SA.3 해당 기능에 대하여 선택 옵션이 있는 경우, 속성은 ‘기능이름’으로 명칭을 정의, 옵션을 속성의 선택 값으로 정의 (CA.3과 같은 방법)

SA.3.1 옵션을 선택하지 않아도 되는 경우에는 ‘none’ 옵션을 정의

SA.3.2 옵션을 여러 가지 선택해도 되는 경우, 선택 가능한 모든 경우를 옵션으로 정의

SA.4 운영체제에 종속적인 프로그램 개발 요소의 값을 설정하기 위해서 종속속성 정의. 속성의 명칭은 개발 요소의 의미를 명확히 알 수 있도록 정의 (CA.4와 같은 방법)

SA.5 각 속성 유형 결정. 종속속성은 각 운영체제 별로 정의

ADC: 데이터 감지와 관련된 속성 유형

SEN: 센서 역할과 관련된 속성 유형

ACT: 엑츄에이터 역할과 관련된 속성 유형

ROU: 라우터 역할과 관련된 속성 유형

SIN: 싱크 역할과 관련된 속성 유형

PSS: 플랫폼에 따라 옵션이 달라지는 속성 유형

PS: ADC, ACT, ROU, SIN, PSS를 제외한 플랫폼에 종속적인 속성 유형

4. 적용 사례

본 논문에서는 적용 사례로써 Nano-Qplus[7]와 TinyOS[8] 두 개의 플랫폼을 위한 속성 설계를 수행하고, 속성의 값을 설정하여 농작물 관리 시스템의 PIM과 PSM을 설계하는 예를 보인다.

[Table 2] Platform function extraction

Function \ Platform	Nano-Qplus	TinyOS
Scheduling	v	v
RF communication	v	v
Real Time Clock		v
Timer	v	v
Sensing(ADC)	v	v
UART	v	v
EEPROM read/write	v	v
Flash Memory read/write	v	
LED	v	v
I2C		v
Hardware ID access		v
WatchDog		v
Hardware potentiometer		v
Encryption/decryption		v
Utility function	v	
System log function	v	
Random number generator		v

Table 2는 시스템 구성을 위한 Nano-Qplus와 TinyOS 플랫폼의 기능 도출의 결과를 보여준다. 두 개의 플랫폼

모두 체크가 된 기능은 공통된 기능이며, 한 쪽에만 체크된 기능은 플랫폼 종속 기능이 된다.

Table 3은 도출한 플랫폼의 공통기능으로부터 PIM을 설계하기 위해 필요한 공통속성을 정의한 결과를 보여준다.

[Table 3] Common attributes

Common functions and program elements	Attribute	Type
Node name	*nodeName*	*COM*
Node identifier	*nodeID*	*COM*
Type of node	*nodeType*	*COM*
Scheduling	*Scheduler_Enable*	*PSS*
RF communication	*RF_Enable*	*PSS*
Timer	*Timer_Enable*	*COM*
ADC (sensing)	*ADC_Enable*	*ADC*
UART	*UART (none, printf, scanf, printf&scanf)*	*COM*
EEPROM read/write	*EEPROM_Enable*	*COM*
LED	*LED_Enable*	*COM*

공통속성 설계 방법을 적용하여 각 기능의 포함여부를 결정하기 위한 속성은 '_Enable'을 붙였으며, 옵션이 있는 속성은 옵션 기능을 옵션명으로 정의하였으며, 공통된 프로그램 요소는 그 내용을 알 수 있도록 명칭을 정의하였다. 또한 각 속성에 대한 유형을 정의하였다.

Table 4는 Nano-Qplus 플랫폼을 위한 종속속성을 설계한 결과를 보여준다. 기능도출을 통해 식별한 종속기능을 바탕으로 공통속성의 설계 방법과 같은 방법을 이용하여 Nano-Qplus 플랫폼에서 실행될 응용 소프트웨어에 대한 PSM 설계를 위한 종속속성을 정의하였다.

[Table 4] A part of platform-specific attributes for Nano-Qplus

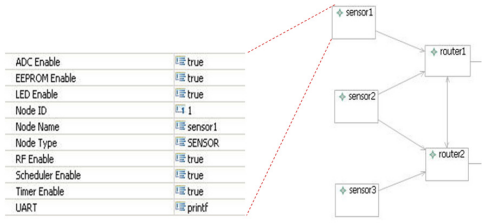
Platform-specific functions and program elements		Attribute	Type
Kind of scheduler		*Scheduler (none, FIFO, PreemptionRR)*	*PSS*
Kind of RF		*RF (Simple, IEEE802.15.4MAC, StarMesh)*	*PSS*
Kind of sensor	Temperature	*Sensor_Temperature_Enable*	*ADC*
	Light	*Sensor_Light_Enable*	*ADC*
	Humidity	*Sensor_Humidity_Enable*	*ADC*
PAN coordinator or not		*isPANCoordinatorNode*	*ROU*
Basic MAC address		*defaultMACAddr*	*PS*
ID of association permission node (start, end node)		*associationPermitStartNodeID, associationPermitEndNodeID*	*PS*
ID of next routing node (first, second node)		*nextHopRoutingFirstNodeID, nextHopRoutingSecondNodeID*	*ROU*

Table 5는 TinyOS 플랫폼을 위한 종속속성을 설계한 결과를 보여준다. Nano-Qplus 플랫폼과 마찬가지로, 기능도출을 통해 식별한 종속기능을 바탕으로 공통속성의 설계 방법과 같은 방법을 이용하여 TinyOS 플랫폼에서 실행될 응용 소프트웨어에 대한 PSM 설계를 위한 종속속성을 정의하였다.

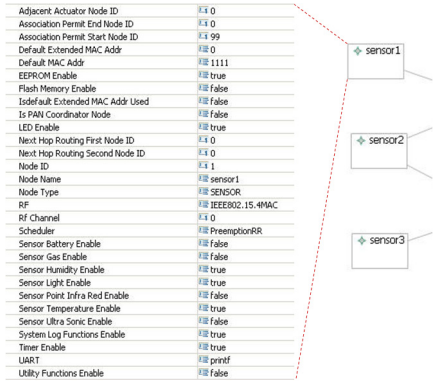
[Table 5] A part of platform-specific attributes for TinyOS

Platform-specific functions and program elements		Attribute	Type
Kind of scheduler		*Scheduler (none, FIFO)*	*PSS*
Kind of RF		*RF (BasicMAC, Broadcast, MultiHop)*	*PSS*
Real Time Clock		*Clock_Enable*	*PS*
TinySec (encryption/decryption)		*TinySec_Enable*	*PS*
Kind of sensor	Temperature	*Sensor_Temperature_Enable*	*ADC*
	Light	*Sensor_Light_Enable*	*ADC*
	Humidity	*Sensor_Humidity_Enable*	*ADC*
	Pressure	*Sensor_Pressure_Enable*	*ADC*
	Microphone	*Sensor_Mic_Enable*	*ADC*

Fig. 2는 Table 3에서 정의한 공통속성을 이용하여 농작물 관리 시스템의 PIM을 설계한 일부를 보여준다. PIM 설계 시에는 그림에서 sensor1 노드와 같이 센서 네트워크를 구성하는 모든 노드에 대하여 공통속성의 값을 설정해 준다.



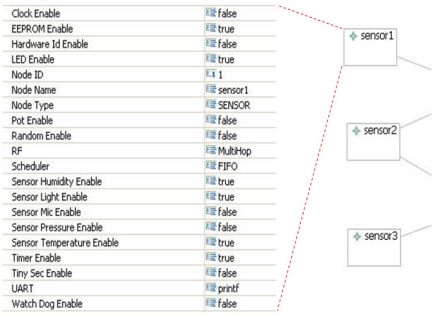
[Fig. 2] Setting of common attributes values for PIM design



[Fig. 3] Setting of platform-specific attributes values for PSM design for Nano-Qplus platform

Fig. 3은 Table 4에서 정의한 Nano-Qplus 플랫폼의 종속속성을 이용하여 농작물 관리 시스템의 PSM을 설계한 일부를 보여준다. 그림에서는 sensor1 노드에 대한 종속속성의 값을 설정한 것을 보여주는데, 이와 같은 방법으로 센서 네트워크 모델에 있는 모든 클래스에 대하여 타겟 플랫폼에 따른 종속속성의 값을 설정함으로써 응용 소프트웨어의 PSM을 설계할 수 있다.

Fig. 4는 Table 5에서 정의한 TinyOS 플랫폼의 종속속성을 이용하여 농작물 관리 시스템을 위한 센서 네트워크 응용 소프트웨어의 PSM을 설계한 일부를 보여준다.



[Fig. 4] Setting of platform-specific attributes values for PSM design for TinyOS platform

```
...

void rf_send_pkt(void)
{
    ...

    int_data = get_light_adc_raw_data();
    sensor_light = int_data;
    byte_data  = (BYTE)LIGHT_SENSOR;
    pBuffer[index] = byte_data;
    index++;
    pBuffer[index] = (BYTE)(0xFF&(int_data>>8));
    index++;
    pBuffer[index] = (BYTE)(0xFF&int_data);
    index++;
    count++;

    unsigned int crt_temp;
    int_data = get_temperature_adc_raw_data();
    crt_temp = (((int_data * 10001) * ADC_REF_VCC) / 10241);
    crt_temp = get_calibration_temp_data(LM61_SENSOR, crt_temp);
    pBuffer[index] = (BYTE)TEMP_SENSOR;
    index++;
    pBuffer[index] = (BYTE)(0xFF&(int_data>>8));
    index++;
    pBuffer[index] = (BYTE)(0xFF&int_data);
    index++;
    count++;

    int_data = get_humidity_fcnt_data();
    pBuffer[index] = (BYTE)HUMIDITY_SENSOR;
    index++;
    pBuffer[index] = (BYTE)(0xFF&(int_data>>8));
    index++;
    pBuffer[index] = (BYTE)(0xFF&int_data);
    index++;
    count++;

    ...
}

int main(void)
{
    uint8_t int_handle;

    int_handle = thread_disable_ints();
    initialize_nano_resources();
    thread_enable_ints(int_handle);

    mlme_ll_link_start(NULL,rf_recv_data);
    pthread_create(NULL,NULL,start,(void *)0);
    start_threads();

    return 0;
}
```

[Fig. 5] Generated source code of sensor1 node software for Nano-Qplus platform

Fig. 5는 Fig. 3과 같이 설계한 Nano-Qplus 운영체제에서 동작하는 센서 노드에 대한 소프트웨어 생성 결과를 보여준다. Fig. 3과 같이 속성 설정을 통하여 특정 운영체제에 대한 PSM을 설계하면, Fig. 5와 같이 각 운영체제가 제공하는 모듈을 조합하여 소프트웨어 코드를 자동으로 생성할 수 있다.

지금까지 본 논문에서 제안하는 속성 설계 기법을 이용한 속성 기반의 센서 네트워크 노드 소프트웨어 개발 방법을 사례 연구로 살펴보았다. 본 논문에서는 사례 연구로 Nano-Qplus와 TinyOS 운영체제에 대해서 속성을 설계하고, 설계한 속성을 기반으로 노드 소프트웨어의 모델을 작성하고 코드를 생성하는 것을 보였지만, 다른 운영체제에 대해서도 같은 방식을 적용하여 노드 소프트웨어를 생성할 수 있다.

적용하고자 하는 센서 운영체제가 있다면, 본 논문에서 제안하는 속성 설계 기법을 이용하여 공통속성과 종속속성을 설계하고, 설계한 속성과 각 운영체제가 제공하는 모듈을 연결시킨 후, 모델 설계 시에 설정한 속성의 값에 따라 해당 모듈을 선택하고 조합함으로써 노드 소프트웨어의 코드를 생성할 수 있다. 센서 노드 소프트웨어는 각 운영체제가 제공하는 모듈을 이용하여 생성되기 때문에 속성과 운영체제의 모듈을 연결시켜 놓으면, 운영체제가 어떤 언어로 개발되어 있든지 운영체제가 개발된 언어로 노드 소프트웨어를 생성할 수 있다.

5. 결론

기존에 센서 네트워크 응용 소프트웨어를 쉽고 빠르게 개발하기 위한 여러 방법들이 연구되었으나, 기존의 방법들은 특정 운영체제 플랫폼을 위한 응용 소프트웨어만 개발 가능하고, 소프트웨어 설계 시 상위 수준의 프로그램을 직접 작성해야 하며, 플랫폼의 기능이 변경되면 프로그램을 다시 작성해야 한다는 한계점을 가지고 있다.

이러한 기존 연구의 한계점을 해결하기 위하여, 본 논문에서는 모델로부터 여러 플랫폼에 맞는 센서 네트워크 응용 소프트웨어를 효율적으로 개발하기 위한 방법으로 MDA의 개념을 적용하고 속성 기반 프로그래밍의 개념을 확장하여 응용 소프트웨어를 개발하는 속성 기반 개발 방법을 지원하기 위한 속성 설계 기법을 제안하였다.

본 논문에서는 PIM과 PSM을 모델링하기 위한 속성을 설계하는 방법을 기술하였으며, 사례 연구로써 제안한 방법을 이용하여 멀티플랫폼을 위한 노드 소프트웨어를 개발하기 위한 속성을 설계하고 설계한 속성을 이용하여 효율적으로 PIM과 PSM을 모델링하는 과정을 보였다.

본 논문에서 제안하는 기법은, 센서 네트워크 응용 소프트웨어의 개발 생산성을 향상시킨다. 센서 네트워크에

는 수많은 노드들이 존재하고 센서 네트워크 응용 소프트웨어는 각 노드마다 개발되어야 하므로, 수많은 응용 소프트웨어를 쉽고 빠르게 개발할 필요가 있다. 제안기법의 속성을 이용하여 다양한 플랫폼을 위한 센서 네트워크 응용 소프트웨어를 쉽고 빠르게 설계할 수 있으므로 효율적이다. 또한, 센서 네트워크 응용 소프트웨어의 유지보수성을 향상시킨다. 센서 네트워크 시스템을 효율적으로 관리하기 위해서는 센서 네트워크의 수행을 위한 수많은 응용 소프트웨어의 유지보수가 중요하다. 제안하는 기법에서는 센서 네트워크 플랫폼에 변화가 생기거나 새로운 플랫폼이 추가되면, 그에 따라 속성을 변경하거나 새로운 속성을 설계하여 추가하고 간단하게 속성 값의 설정을 통하여 응용 소프트웨어 코드를 재생성할 수 있으므로 유지보수성이 향상된다.

References

[1] I. F. Akyildiz, W. L. Su, Y. Sankarasubramaniam, and E. Cayirci, “A survey on sensor networks,” IEEE Communications Magazine, vol. 40, no. 8, pp102-114, Aug. 2002.
DOI: http://dx.doi.org/10.1109/MCOM.2002.1024422

[2] M. Welsh, G. Mainland, “Programming sensor networks using abstract regions,” In Proceedings of the 1st USENIX/ACM Symposium on Networked Systems Design and Implementation (NSDI’04), 2004.

[3] L. Mottola, G. P. Picco, “Logical neighborhoods: A programming abstraction for wireless sensor networks,” In Proceedings of the 2nd International Conference on Distributed Computing in Sensor Systems(DCOSS’06), 2006.

[4] S. Madden, M. J. Franklin, J. M. Hellerstein, “The design of an acquisitional query processor for sensor networks,” In Proceedings of the ACM SIGMOD International Conference on Management of Data (SIGMOD’03), 2003.
DOI: http://dx.doi.org/10.1145/872757.872817

[5] L. Mottola, G. P. Picco, “Programming Wireless Sensor Networks with Logical Neighborhoods: A Road Tunnel Use Case,” In Proceedings of the 5th ACM International Conference on Embedded Networked Sensor Systems (SENSYS07), 2007.
DOI: http://dx.doi.org/10.1145/1322263.1322311

[6] R. Gummadi, O. Gnawali, R. Govindan, “Macro-programming wireless sensor networks using Kairos,” In Proceedings of the 1st International Conference on Distributed Computing in Sensor Systems (DCOSS’05), 2005.

[7] K. Lee, Y. Shin, H. Choi, S. Park, “A Design of Sensor Network System based on Scalable & Reconfigurable Nano-OS Platform,” In Proceedings of the IT SoC Conference, 2004.

[8] J. Hill, R. Szewczyk, A. Woo, S. Hollar, D. Culler,Kristofer Pister, “System architecture directions for network sensors,” In Proceedings of the 9th International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS), 2000.

[9] C. Han, R. Rengaswamy, R. Shea, E. Kohler, M.Srivastava, “SOS: A dynamic operating system for sensor networks,” In Proceedings of the Third International Conference on Mobile Systems, Applications, And Services (Mobisys), 2005.

[10] S. Bhatti, J. Carlson, H. Dai, J. Deng, J. Rose, A.Sheth, B. Shucker, C. Gruenwald, A. Torgerson, R.Han, “MANTIS OS: An Embedded Multithreaded Operating System for Wireless Micro Sensor Platforms,” ACM/Kluwer Mobile Networks & Applications, Special Issue on Wireless Sensor Networks, vol. 10, no. 4, pp. 563-579, 2005.
DOI: http://dx.doi.org/10.1007/s11036-005-1567-8

[11] A. Dunkels, B. Grönvall, T. Voigt, “Contiki - a Lightweight and Flexible Operating System for Tiny Networked Sensors,” In Proceedings of the First IEEE Workshop on Embedded Networked Sensors (EmNets), 2004.
DOI: http://dx.doi.org/10.1109/LCN.2004.38

[12] L. Gu, J. Stankovic, “t-kernel: Provide Reliable OS Support for Wireless Sensor Networks,” In Proceedings of the 4th ACM Conference on Embedded Networked Sensor Systems (Sensys), 2006.
DOI: http://dx.doi.org/10.1145/1182807.1182809

[13] Q.Cao, T. Abdelzaher, J. Stankovic, T. He, “The LiteOS Operating System: Towards Unix-Like Abstractions for Wireless Sensor Networks," In Proceedings of the 7th International Conference on Information Processing in Sensor Networks, 2008.

[14] A. Eswaran, A. Rowe, R. Rajkumar, “Nano-RK: an energy -aware resource-centric RTOS for sensor networks,” In Proceedings of the 26th IEEE International Real-Time Systems Symposium (RTSS), 2005.
DOI: http://dx.doi.org/10.1109/RTSS.2005.30

[15] A. Kleppe, J. Warmer, W. Bast, The Model Driven

Architecture: Practice and Promise, Addison-Wesley, 2003.
[16] “attribute-based programming,” http://webopedia.com/TERM/A/attribute_based_programming.html
[17] G. Wasson, M. Humphrey, Attribute-based programming for grid services, In Proceedings of the GGF9 Workshop on Designing and Building Grid Services, 2003.
[18] “Visual C++ Attributed Programming Concepts," http://msdn.microsoft.com/en-us/library/zkwy014e(VS.80).aspx

이 우 진(Woo-Jin Lee) **[정회원]**

- 2002년 2월 : 숭실대학교대학원 (컴퓨터공학석사)
- 2007년 2월 : 숭실대학교대학원 (컴퓨터공학박사)
- 2009년 3월 ~ 현재 : 세종대학교 정보통신공학과 초빙교수

<관심분야>
유비쿼터스 컴퓨팅, 모바일 컴퓨팅, 소프트웨어개발

최 일 우(Il-Woo Choi) **[정회원]**

- 1997년 2월 : 숭실대학교대학원 (컴퓨터공학석사)
- 2004년 2월 : 숭실대학교대학원 (컴퓨터공학박사)
- 2007년 3월 ~ 현재 : 강남대학교 교양학부 교수

<관심분야>
개발 프로세스, 재사용, SOA, USN, 모바일 컴퓨팅

김 주 일(Ju-Il Kim) **[정회원]**

- 2006년 2월 : 숭실대학교대학원 (컴퓨터공학석사)
- 2010년 2월 : 숭실대학교대학원 (컴퓨터공학박사)
- 2011년 8월 ~ 현재 : ㈜지아이티 GDS진단팀 책임연구원

<관심분야>
유비쿼터스 컴퓨팅, 임베디드 시스템, 모바일 컴퓨팅